14
Inheritance Extending Classes

14.1 Introduction

This lesson discusses about inheritance, the capability of one class to inherit properties from another class as a child inherits some properties from his/her parents. The most important advantage of inheritance is code reusability. Once a base class is written and debugged, it can be used in various situations without having to redefine it or rewrite it. Reusing existing code saves time, money and efforts of writing the code again. Without redefining the old class, you can add new properties to desired class and redefine an inherited class member function.

14.2 Objectives

After going through this lesson, you would be able to

- explain the concept of inheritance
- describe the five forms of inheritance
- define three types of inheritance
- explain all three visibility modes
- describe the concept of abstract class & virtual class

14.3 Need for Inheritance

Inheritance is one of the important concepts of object-oriented language. There are several reasons why this concept was introduced in object oriented language. Some major reasons are:

(i) The capability to express the inheritance relationship which ensures the closeness with the real world model.

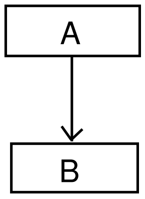
(ii) Idea of reusability, i.e., the new class can use some of the features of old class.

(iii) Transitive nature of inheritance, i.e., it can be passed on further.

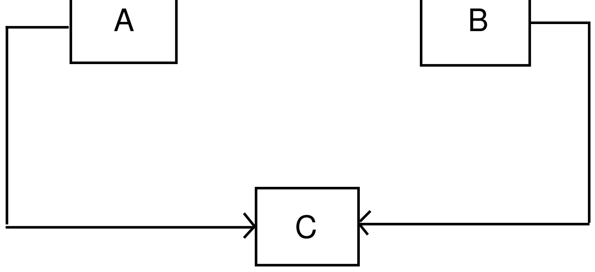
14.4 Different Forms of Inheritance

The mechanism of deriving a new class from an old one is called inheritance (or derivation). The old class is referred to as the base class and new one is called the derived class. There are various forms of inheritance.

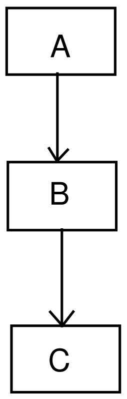
(i) Single inheritance —> A derived class with only one base class is called single inheritance.



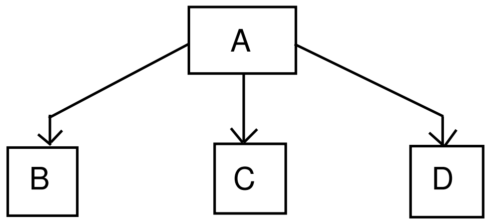
(ii) Multiple inheritance —> A derived class with several base classes is called multiple inheritance.



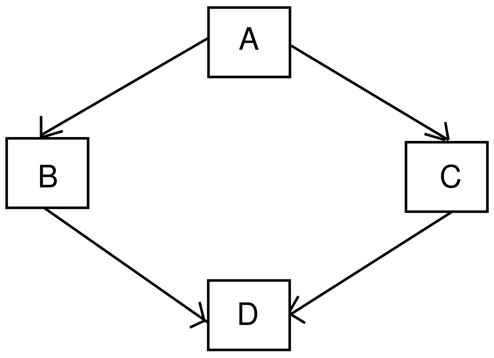
(iii) Multilevel inheritance —> The mechanism of deriving a class from another derived class is called multilevel inheritance.



(iv) Hierarchical inheritance —> One class may be inherited by more than one classes. This process is known as hierarchical inheritance.



(v) Hybrid inheritance —> It is a combination of hierarchical and multiple inheritance.



14.5 Defining Derived Class

A derived class is defined by specifying its relationship with the base class using visibility mode.

The general form of defining a derived class is:

```
class derived_class : visibility_mode base_class

{

_________________

__________________ // members of derived class.

};
```

The colon indicates that the derived_class is derived (inherits some property) from base_class.

The base class(es) name(s) follow(s) the colon (:). The names of all the base classes of a derived class follow : (colon) and are separated by comma. The visibility-mode can be either private or public or protected. If no visibility mode is specified, then by default the visibility mode is considered as private.

Following are some examples of derived class definitions:

```
class Marksheet : public student / / public derivation
{
    // members of derived class
};
class Marksheet : private student / / private derivation
    // members of derived class
};
class Marksheet : protected student // protected derivation
{
    // members of protected class
};
```

In the above definitions, Marksheet is the derived class of student base class. The visibility mode public indicates that student is a public base class. Similarly, the visibility modes private or protected indicates that student is private base class or protected base class respectively.

When we say that members of a class are inheritable, it means that the derived class can access them directly. However, the derived class has access privilege only to the non-private members of the base class. Although the private members of the base class cannot be accessed directly, yet the objects of derived class are able to access them through the non-private inherited members.

14.6 Multiple Inheritance

As we know that a subclass inheriting from multiple base classes is known as Multiple inheritance. The Syntax of defining a derived class is given below :

```
class derived_classname : mode baseclass1, mode baseclass2
{
// members of derived class
};
```

Example 1

```
class marks : public semester1, private semester2
{
    // members
};
```

14.7 Visibility Modes

It can be public, private or protected. The private data of base class cannot be inherited.

(i) If inheritance is done in public mode, public members of the base class become the public members of derived class and protected members of base class become the protected members of derived class.

(ii) In inheritances is done in a private mode, public and protected members of base class become the private members of derived class.

(iii) If inheritance is done in a protected mode, public and protected members of base class become the protected members of derived class.

The following table shows the three types of inheritance:

Base class Access specifier	Derived class public	private	protected
public	public	private	protected
private	Not inherited	Not inherited	Not inherited
protected	protected	protected	Protected

The Public Visibility mode

The following example and figure illustrate the public derivation in classes.

```
class student
{
private :
      int x;
      void getdata (  );
public:
      int y;
      void putdata ( );
protected:
      int z;
      void check ( );
};
class marks : public student
{
private :
      int a ;
      void readdata (  );
public :
      int b;
      void writedata ( );
protected :
    int c;
void checkvalue ( );
};
```

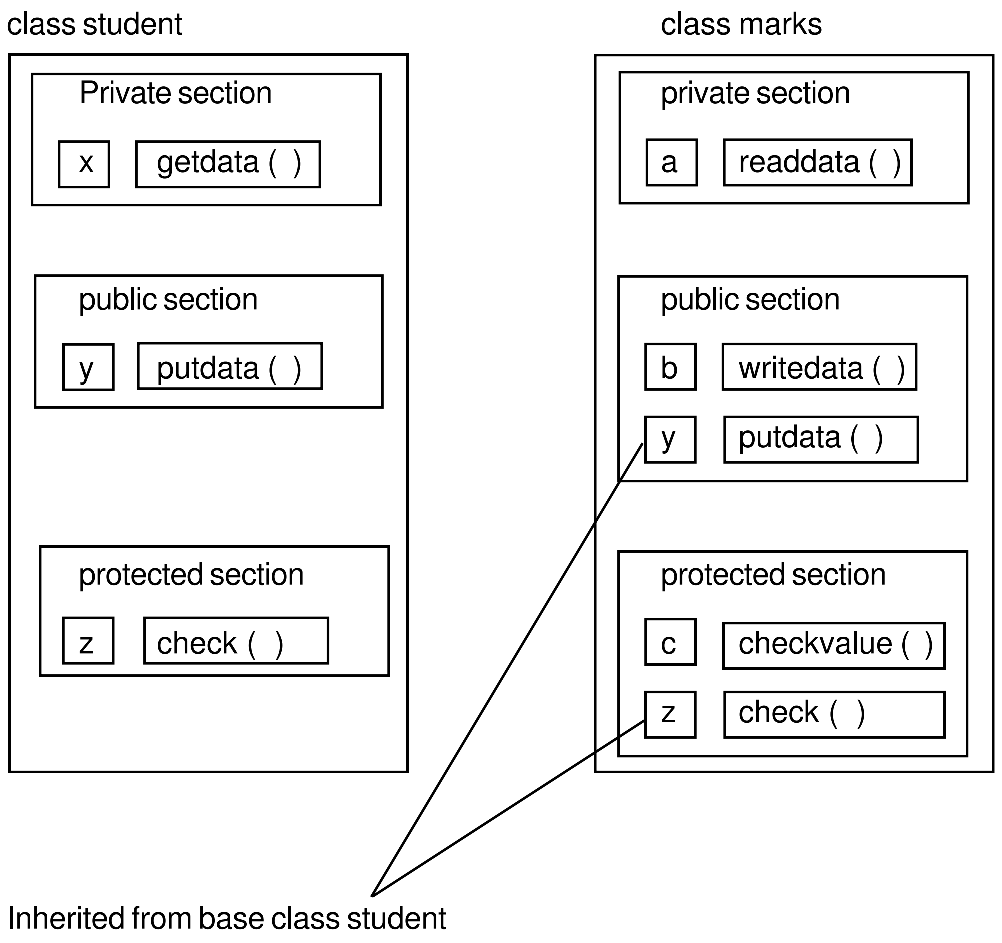


Fig. 14.1 public derivation of a class

The public derivation does not change the access specifiers for inherited members in the derived class. The private data of base class student cannot be inherited.

The Private Visibility Mode

We are using the same example, but the derivation is done in private mode.

```
Class student

{

// same as in previous example
};
class marks : private student
{
    //
};
```

The following figure illustrates the private derivation in the classes.

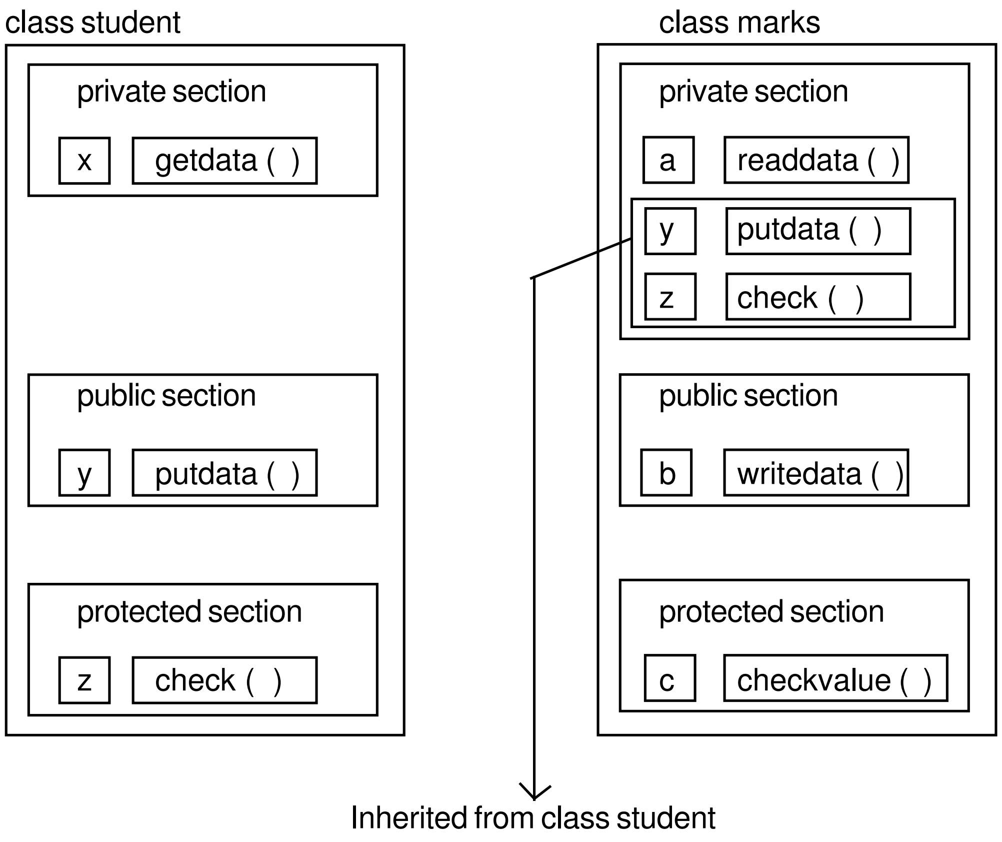


Fig 14.2 Private derivation of a class

As it is clear from the figure that the data present in public and protected section of base class become the private members of derived class. The data in private section of base class cannot be inherited.

The Protected visibility mode

We are using the same example but the derivation is done in protected mode.

```
class student
{
```

```
// same as in previous example
};
class marks : protected student
{
};
```

The following figure illustrates the protected derivation in the classes.

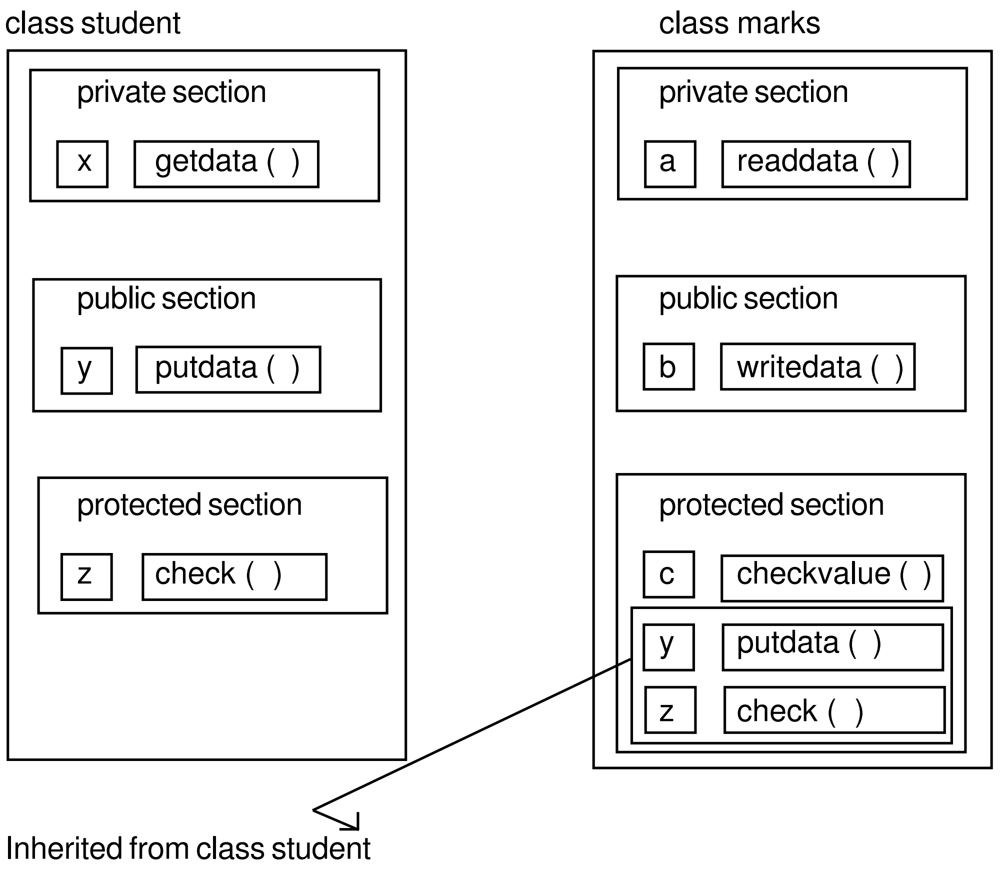


Fig. 14.3 Protected derivation of a class

The data present in private section of base class cannot be inherited. The difference between private and protected section is that data present in protected section can be inherited. Otherwise both the section cannot be accessed by the object of the class.

14.8 Abstract Class

An abstract class is one that is not used to create objects. An abstract class is designed only to act as a base class (to be inherited by other classes). It is a design concept in program development and provides a base upon which other classes may be built. In the previous example, the class student is an abstract class since it was not used to create any object.

14.9 Virtual Base Class

Consider a situation where all the three kinds of inheritance, namely, multilevel, multiple and hierarchical instances are involved. Consider the following example illustrated in a figure.

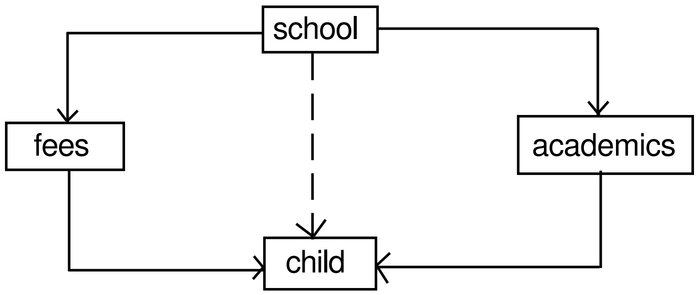


Fig. 14.4

The child has two direct base classes fees and academics which themselves have a common base class 'school'. The child inherits the traits of 'school' via two separate paths. It can also inherit directly as shown by the broken line. The 'school' is sometimes referred to as indirect base class. All the public and protected members of 'school' are inherited into 'child' twice, first via 'fees' and again via 'academics'. This means, 'child' would have duplicate sets of the members inherited from 'school'. This introduces ambiguity and should be avoided.

The duplication of inherited members due to these multiple paths can be avoided by making the common base class as virtual class by declaring the base class as shown below :

```
class school
{    --------------
     ----------------
};
class fees : virtual public school
{    ---------------
     -----------------
};
```

```
class academics : public virtual school
{   -------------------
    --------------------
};
class child : public fees, public academics
{
    / / only one copy of school will be inherited.
};
```

Note : you can write either virtual public or public virtual.

In-Text Questions 14.1

1. What is inheritance? What are base class and derived class ?
2. What are the different forms of inheritance?
3. What are the three modes of inheritance?
4. The private section of base class can be inherited. True or False?
5. What is the difference between private and protected sections ?
6. What is an abstract class?
7. Fill in the blanks:
   (a) The base class is also called ....
   (b) The derived class can be derived from base class in .... or .... way.
   (c) By default base class is visible as .... mode in derived class.
   (d) When a derived class is derived from more than one base class then the inheritance is called .... inheritance.
8. State whether the following are True or False.
   (a) Inheritance means child receiving certain traits from parents.
   (b) The default base class is visible as public mode in derived class.

(c) When a derived class is derived from more than one base class then the inheritance is called hierarchical inheritance.

(d) The base class is called abstract class

(e) Private data of base class can be inherited

9. Consider the following class declaration and answer the questions given below:

```
class zoo
{
     char location [20];
     protected;
        int no_of_animals;
public;
     void inputdata (char, int);
     vod outputdata ( );
};
     class animal : protected zoo
{
     int tail;
     protected;
        int legs;
public:
     void readdata (int, int) ;
     void writedata ( );
};
class carnivorous : private animal
{
     int paw_size;
     public :
        void fetchdata (int);
        void displayed ( ) ;
};
```

(a) Name the base class and derived class of the class animal.

(b) Name the data member(s) that can be accessed from function displayed ( ).

(c) Name the data member(s) that can be accessed by an object of carnivorous class.

(d) Is the member function outputdata accessible to the objects of animal class ?

14.10 What you have learnt

In this lesson, you learnt about a very important aspect of object oriented programming i.e., inheritance. We discussed different forms and types of inheritance so that you should be able to use inheritance in C++ programming.

14.11 Terminal Questions

1. What are the needs of inheritance?

2. What are the three modes of inheritance? Explain.

3. What is a virtual base class ? Explain it by taking an example.

4. Consider the following class declaration and answer the questions given below:

```
class vehicle
{
    int wheels;
    protected;
       int passenger;
public;
    void inputdata (int, int);
    void outputdata ( );
};
class heavy_vehicle: protected vehicle
{
    int diesel_petrol;
    protected:
       int load;
public:
    void readdata (int, int) ;
    void writedata ( );
};
class bus : private heavy_vehicle
{
    char make [20];
    public :
        void fetchdata (int);
        void displaydata ( ) ;

};
```

(i) Name the base class and derived class of the class heavy_vehicle.

(ii) Name the data member(s) that can be accessed from function displaydata.

(iii) Name the data member(s) that can be accessed by an object of bus class?

(iv) Is the member function outputdata accessible to the objects of heavy_vehicle class?

14.12 Feedback to In-Text Questions

In-text Questions 14.1

1. The mechanism of deriving a new class from an old one is called inheritance. The old class is referred to as the base class and new one is called the derived class.

2. (i) Single inheritance

   (ii) Multiple inheritance

   (iii) Multilevel inheritance

   (iv) Hybrid inheritance

   (v) Hierarchical inheritance

3. private, public and protected

4. False

5. The data in protected section can be inherited.

6. The base class is called an abstract class.

7. (a) Abstract class
   (b) public, private, protected
   (c) private
   (d) multiple

8. (a) T
   (b) F
   (c) F
   (d) T
   (e) F

9. (a) base class - zoo
   derived class - carnivorous
   (b) legs, no-o-animals, paw_size
   (c) None
   (d) No